


```
RRRRRRRR  MM      MM  SSSSSSSS  000000  SSSSSSSS  RRRRRRRR  CCCCCCCC  HH      HH
RRRRRRRR  MM      MM  SSSSSSSS  000000  SSSSSSSS  RRRRRRRR  CCCCCCCC  HH      HH
RR      RR  MMMM  MMMM  SS      00      00  SS      RR      RR  CC      HH      HH
RR      RR  MMMM  MMMM  SS      00      00  SS      RR      RR  CC      HH      HH
RR      RR  MM  MM  MM  SS      00      0000  SS      RR      RR  CC      HH      HH
RRRRRRRR  MM      MM  SSSSSS  00      00  0000  SSSSSS  RRRRRRRR  CC      HH      HH
RRRRRRRR  MM      MM  SSSSSS  00      00  0000  SSSSSS  RRRRRRRR  CC      HHHHHHHHHH
RR      RR  MM      MM  SS      0000      00  SS      RR      RR  CC      HH      HH
RR      RR  MM      MM  SS      0000      00  SS      RR      RR  CC      HH      HH
RR      RR  MM      MM  SS      00      00  SS      RR      RR  CC      HH      HH
RR      RR  MM      MM  SSSSSSSS  000000  SSSSSSSS  RR      RR  CC      HH      HH
RR      RR  MM      MM  SSSSSSSS  000000  SSSSSSSS  RR      RR  CCCCCCCC  HH      HH
                                           CCCCCCCC  HH      HH
                                           .....
                                           .....
                                           .....
                                           .....

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```

(3)	99	DEFINITIONS
(4)	119	RMS\$SEARCH, Search for next Filename in Sequence
(6)	559	RMSCOPY RESULT, Return Result Name String
(7)	613	RETDIRBDB, Deallocate Directory Buffer and BDB

```
0000 1          $BEGIN RMSOSRCH,000,RMSRMS,<SEARCH FOR NEXT WILDCARD FILE>
0000 2
0000 3
0000 4 *****
0000 5 *****
0000 6 *
0000 7 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 *  ALL RIGHTS RESERVED.
0000 10 *
0000 11 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 *  TRANSFERRED.
0000 17 *
0000 18 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 *  CORPORATION.
0000 21 *
0000 22 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *****
0000 26 *****
```



```
0000 28 :++
0000 29 :
0000 30 : Facility: rms32
0000 31 :
0000 32 : Abstract:
0000 33 :     this is the highest level routine to perform the
0000 34 :     $remove and $search functions
0000 35 :
0000 36 : Environment:
0000 37 :     vax/vms
0000 38 :
0000 39 : Author:
0000 40 :     tim halvorsen   AUG-1979
0000 41 :
0000 42 : Modified By:
0000 43 :
0000 44 :     V03-014 JEJ0026      J E Johnson      11-Apr-1984
0000 45 :     Tie off invalid network operations.
0000 46 :
0000 47 :     V03-013 DGB0022      Donald G. Blair    06-Mar-1984
0000 48 :     Use full-length FIB to support access mode protected
0000 49 :     files. Also change RMSRETDIRBDB to RETDIRBDB, a local
0000 50 :     routine.
0000 51 :
0000 52 :     V03-012 RAS0219      Ron Schaefer      8-Dec-1983
0000 53 :     Change references to FWAST_SWB subfield to separate structure.
0000 54 :
0000 55 :     V03-011 RAS0201      Ron Schaefer      17-Oct-1983
0000 56 :     Correct calls to RMSPARSE_FILE to account for the fact
0000 57 :     that it does NOT necessarily preserve R7.
0000 58 :     Make sure we got a name, type and/or version from ESA/ESL.
0000 59 :
0000 60 :     V03-010 KBT0585      Keith B. Thompson  12-Aug-1983
0000 61 :     Cleanup fwa constants
0000 62 :
0000 63 :     V03-009 KBT0559      Keith B. Thompson  20-Jul-1983
0000 64 :     Convert DNF and FNF errors into NMF after a sucessful
0000 65 :     search list operation
0000 66 :
0000 67 :     V03-008 KBT0533      Keith B. Thompson  1-Jun-1983
0000 68 :     Turn on search list processing and remove ref. to
0000 69 :     RM$SKIP_SUBTREE (this was a JSB to a SSB!)
0000 70 :
0000 71 :     V03-007 RAS0122      Ron Schaefer      1-Feb-1983
0000 72 :     Complete KBT0472 by correcting a problem that would
0000 73 :     leave an IFAB marked busy if the saved NAM block IF1
0000 74 :     was incorrect.
0000 75 :
0000 76 :     V03-006 KBT0472      Keith B. Thompson  24-Jan-1983
0000 77 :     Fix some code i don't understand
0000 78 :
0000 79 :     V03-005 RAS0103      Ron Schaefer      19-Nov-1982
0000 80 :     Correct saving of the caller's access mode so that
0000 81 :     exits via RMSEX_NOSTR have the caller's mode in R7;
0000 82 :     and correct DMW4004 to correctly save the mode in the IFB.
0000 83 :
0000 84 :     V03-004 DMW4004      DMWalp            2-Sep-1982
```

0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :--
0000 96 :
0000 97 :

Added code so that RMSSFABCHK was not called twice;
it was called directly and then again in RMSSFSETI

V03-003 KBT0194 Keith B. Thompson 23-Aug-1982
Reorganize psects

V03-002 KRM0058 K Malik 10-Aug-1982
Changed FWASB_UNDERLINE symbol to FWASB_UNDER_DEV
(to distinguish it from new FWASB_UNDER_NOD symbol).

```
0000 99 .SBTTL DEFINITIONS
0000 100
0000 101 ::
0000 102 :: symbol definitions
0000 103 ::
0000 104
0000 105 $IODEF
0000 106 $RMSDEF
0000 107 $SSDEF
0000 108 $BDBDEF
0000 109 $DEVDEF
0000 110 $FABDEF
0000 111 $FIBDEF
0000 112 $FWADEF
0000 113 $IFBDEF
0000 114 $IMPDEF
0000 115 $NAMDEF
0000 116 $FSCBDEF
0000 117 $SWBDEF
```



```
0000 119 .SBTTL RMS$SEARCH, Search for next Filename in Sequence
0000 120
0000 121 :++
0000 122 :
0000 123 RMS$SEARCH
0000 124 :
0000 125 Search for next filename in sequence
0000 126 :
0000 127 RMS$REMOVE
0000 128 :
0000 129 Remove a directory entry
0000 130 :
0000 131 inputs:
0000 132 :
0000 133 ap = address of user argument list
0000 134 wcc of nam block contains ifi of wildcard ifab
0000 135 :
0000 136 outputs:
0000 137 :
0000 138 result name string is returned to user buffer
0000 139 fid/did in nam block
0000 140 :
0000 141 :--
0000 142 :
0000 143 $ENTRY RMS$REMOVE
7E 35 9A 0000 144 MOVZBL #IOS_DELETE,-(SP) ; set acp function code = remove
0000 145 BRB COMMON
0000 146 :
0000 147 $ENTRY RMS$SEARCH
7E 32 9A 0005 148 MOVZBL #IOS_ACCESS,-(SP) ; set acp function code = search
0000 149 : ; this cannot be popped until ret
0000 150 : ; since rm$iset saves the sp for stall
0000 151 COMMON: STSTPT SEARCH
000E 152 :
000E 153 :
000E 154 : Get ifab and fwa addresses from ifi which resides in wcc
000E 155 : of nam block.
000E 156 :
000E 157 :
FFEF* 30 000E 158 BSBW RMS$FABCHK ; check fab validity returns only if ok
0011 159 : ; r11 = impure area
0011 160 : ; r8 = fab address
0011 161 : ; r7 = caller's access mode
0011 162 BEQL 10$ ; check IFI
0013 163 RMSERR IFI ; error if IFI non-zero
0018 164 BRB 20$
001A 165 :
57 28 A8 DD 001A 166 10$: PUSHL R7 ; save caller's mode
FFDD* 30 001C 167 MOVL FAB$L_NAM(R8),R7 ; get nam address
56 57 D0 0020 168 BSBW RMS$CHRNAM ; check nam validity
57 8E D0 0023 169 MOVL R7,R6 ; copy nam addr
12 50 E9 0026 170 MOVL (SP)+,R7 ; restore caller's mode
0029 171 BLBC R0,20$ ; if error
002C 172 : ; take the 'nostruct' error exit
002C 173 BITW NAM$L_WCC+2(R6),- ; check to see that no spurious bits
3FFE 8F 002F 174 #^C<<NAM$M_SVCTX!- ; other than the IFI bit, the search
0032 175 NAM$M_SRCHNMF- ; NMF bit, or the save context bit are
```



```
0032 176 @-16>!1> ; set within the field NAMS_L_WCC
66 12 0032 177 BNEQ ERRWCC ; error if illegal wcc value
1E E1 0034 178 BBC #NAMS_V SRCHNMF,- ; if NMF has been encountered,
07 30 A6 0036 179 NAMS_L_WCC(R6),30$ ; then go immediately return
5F 11 0039 180 RMSERR NMF ; a status of NMF
20$: BRB ENS
0040 182
59 30 A6 3C 0040 183 30$: MOVZWL NAMS_L_WCC(R6),R9 ; get ifi of previous ifab
1E 13 0044 184 BEQL 50$ ; branch if none
0046 185
10 E1 0046 186 BBC #NAMS_V IFI,- ; if the IFI bit is not set then
19 30 A6 0048 187 NAMS_L_WCC(R6),50$ ; context has not been saved
02 AB 59 B0 004B 188 MOVW R9,FAB$W_IFI(R8) ; set internal ifi into fab
FFAE' 30 004F 189 BSBW RMSFSET_ALT1 ; setup with ifi in fab
14 AB 04 C0 0052 190 ADDL2 #4,IMP$[SAVED_SP(R11) ; adjust FSET saved sp for acp code
06 69 39 E1 0056 191 BBC #IFB$V_SEARCH,(R9),40$ ; branch if not our type of ifab
5A 38 A9 D0 005A 192 MOVL IFB$L_FWA_PTR(R9),R10 ; get fwa
1B 12 005E 193 BNEQ SRCH ; branch if have one
40$: CSB #IFB$V_BUSY,(R9) ; don't leave this IFAB marked busy
0060 194
0064 195
0064 196 ;
0064 197 ; No previous context can be found, parse the expanded name
0064 198 ; string and proceed.
0064 199 ;
0064 200
FF99' 30 0064 201 50$: BSBW RMSFSETI_ALT ; allocate ifab/ifi
14 AB 04 C0 0067 202 ADDL2 #4,IMP$[SAVED_SP(R11) ; adjust FSET saved sp for acp code
57 28 AB D0 006B 203 MOVL FAB$L_NAM(R8),R7 ; get nam address
FF8E' 30 006F 204 BSBW RMSCHRNAM ; check nam validity
22 50 E9 0072 205 BLBC R0,EXIT1 ; quit on failure
FF88' 30 0075 206 BSBW RMSRECOVER_FWA ; recover fwa context
1C 50 E9 0078 207 BLBC R0,EXIT1 ; branch if error
007B 208
```

```
007B 210 :
007B 211 : Context has been recovered. Check device characteristics. continue only
007B 212 : for directory structured devices.
007B 213 :
007B 214 :
23 6A 19 E0 007B 215 SRCH: BBS #FW$V_NODE,(R10),NTSRCH; branch if network operation
      03 E1 007F 216 BBC #DEV$V_DIR,- ; error if illegal device
      OF 69 0081 217 IFB$V_PRIM_DEV(R9),ERRIOP
      06 E0 0083 218 BBS #DEV$V_SPL,- ; error if spooled device
09 008C C9 0085 219 IFB$V_AS_DEV(R9),ERRIOP
      0089 220
      0089 221 :
      0089 222 : Get the next file in sequence
      0089 223 :
      0089 224 :
01FE CA B5 0089 225 TSTW FFAST_FIBBUF+FIB$V_DID(R10) ; new directory needed?
      47 12 008D 226 BNEQ READ_DIR ; branch if not
016F 31 008F 227 BRW NEXT_DIR ; and get next directory
      0092 228
      0092 229 ERRIOP: RMSERR IOP ; illegal device type
025B 31 0097 230 EXIT1: BRW EXIT ; exit cleaning up ifab
      009A 231
      009A 232 ERWCC: RMSERR WCC ; illegal wcc value
      FF5E' 31 009F 233 ENS: BRW RM$EX_NOSTR ; exit without ifab with status
      00A2 234
      00A2 235 :
      00A2 236 : Perform network search function.
      00A2 237 :
      00A2 238 :
EC 69 3F E0 00A2 239 NTSRCH: BBS #IFB$V_NSP,(R9),ERRIOP ; search of node::"task=abc" is invalid
      00A6 240
      00A6 241 CMPB #IOS_DELETE,(SP) ; Is this a search or a remove op?
      6E 35 91 00A9 242 BNEQ SS ; Branch if a search operation
      05 12 00AB 243 BSBW NTS$REMOVE ; its a remove...
      FF52' 30 00AE 244 BRB EXIT1 ; branch aid to home
      E7 11 00B0 245
      00B0 246 SS: CLRL R0 ; clear first-time-thru flag
06 69 50 D4 00B0 247 BBS #IFB$V_ACCESSED,(R9),10$ ; branch if already connected to fal
      25 E0 00B2 248 BSBW NTS$ACCESS ; establish logical link with fal
      FF47' 30 00B6 249 BLBC R0,20$ ; branch on failure
      OD 50 E9 00B9 250 ; note, first-time-thru flag is now set!
      00BC 251 10$: BSBW NTS$SEARCH ; perform search at remote node
      FF41' 30 00BC 252 BLBC R0,30$ ; branch on failure
      11 50 E9 00BF 253 SSB #IFB$V_FILEFOUND,(R9) ; indicate at least one file found
      01C0 31 00C6 254 BRW COPY_RESULT ; branch aid
      00C9 255 20$: RMSERR FND,R1 ; set default error code
      FF2F' 30 00CE 256 BSBW RM$MAPERR ; map ss error to rms error if possible
      C4 11 00D1 257 BRB EXIT1 ; branch aid
      0147 31 00D3 258 30$: BRW ERROR ; branch aid
      00D6 259
      00D6 260 :
      00D6 261 : If we are saving context (ifab/fwa) and we are searching a wildcard
      00D6 262 : specification and no directory file has been read yet, then read the
      00D6 263 : directory file into memory to optimize on obtaining file names.
      00D6 264 :
      00D6 265 :
      00D6 266 READ_DIR:
```



```
58 24 A9 D0 00D6 267      MOVL  IFBSL_LAST FAB(R9),R8      ; get fab address
69 39 E1 00DA 268      BBC   #IFBSV_SEARCH,(R9),-      ; branch if not saving context
    16 00DD 269      NEXT FILE
6A 18 E1 00DE 270      BBC   #FWASV_WILDCARD,(R10),-      ; branch if non-wild string
    12 00E1 271      NEXT FILE
30 AA D5 00E2 272      TSTL  FWASC_DIRBDB(R10)          ; directory file read yet?
    0D 12 00E5 273      BNEQ  NEXT FILE                  ; if so, don't read again
00000000'EF 16 00E7 274      JSB  RMSREADDIR              ; read directory into memory
    04 50 E9 00ED 275      BLBC  R0,NEXT_FILE             ; branch if unable to read
30 AA 57 D0 00F0 276      MOVL  R7,FWASC_DIRBDB(R10)      ; save bdb address
    00F4 277
    00F4 278      ; Get the file name pattern from the expanded name string
    00F4 279
    00F4 280
    00F4 281
    00F4 282      NEXT_FILE:
57 28 AB D0 00F4 283      MOVL  FABSL_NAM(R8),R7          ; and recover nam address again
    FF05' 30 00F8 284      BSBW  RMSCHRNAM                ; check nam validity
6A 50 E9 00FB 285      BLBC  R0,EXIT2                    ; quit on failure
56 0B A7 9A 00FE 286      MOVZBL NAM$B_ESL(R7),R6          ; length of expanded string
    5F 13 0102 287      BEQL  ERRESL                     ; error if none
57 0C A7 D0 0104 288      MOVL  NAM$B_ESA(R7),R7          ; address of expanded string
    0108 289      IFNORD  R6,(R7),ERR$SA                 ; error if cannot read buffer
52 0104 8F 3C 010E 290      MOVZWL #FSCB$C_BLN,R2          ; get size of FSCB
    FEEA' 30 0113 291      BSBW  RMSGETSPC1              ; allocate it
    4F 50 E9 0116 292      BLBC  R0,EXIT2                ; exit on error
    5B DD 0119 293      PUSHL  R11                        ; save impure area
    5B 51 D0 011B 294      MOVL  R1,R11                    ; put FSCB in correct reg
00000000'EF 16 011E 295      JSB  RMS$SCAN_STRING          ; scan the string
50 2C AB 7D 0124 296      MOVQ  FSCB$Q_NAME(R11),R0        ; get name
    0C 12 0128 297      BNEQ  10$                          ; got one
50 34 AB 7D 012A 298      MOVQ  FSCB$Q_TYPE(R11),R0        ; how about type
    0A 12 012E 299      BNEQ  20$                          ; got one
50 3C AB 7D 0130 300      MOVQ  FSCB$Q_VERSION(R11),R0     ; try version
    08 11 0134 301      BRB    30$                          ; exit
50 34 AB A0 0136 302 10$: ADDW2 FSCB$Q_TYPE(R11),R0        ; add type
50 3C AB A0 013A 303 20$: ADDW2 FSCB$Q_VERSION(R11),R0    ; add version
0188 CA 50 B0 013E 304 30$: MOVW  R0,FWASQ_RNS(R10)        ; set string descriptor length (no flags)
018C CA 51 D0 0143 305      MOVL  R1,FWASQ_RNS+4(R10)      ; and address
    54 5B D0 0148 306      MOVL  R11,R4                    ; get ready to return
    5B 8ED0 014B 307      POPL  R11                        ; restore impure area
52 0104 8F 3C 014E 308      MOVZWL #FSCB$C_BLN,R2          ;
    FEEA' 30 0153 309      BSBW  RMSRETSPC1              ; return FSCB
0188 CA B5 0156 310      TSTW  FWASQ_RNS(R10)              ; valid string for ACP?
    OF 12 015A 311      BNEQ  SETFIB
    015C 312
    015C 313      ERRESA: RMSERR  ESA                      ; set esa error
    05 11 0161 314      BRB    EXIT2
    0163 315
    0163 316      ERRESL: RMSERR  ESL                      ; set esl error
018A 31 0168 317      EXIT2: BRW    EXIT
    016B 318
    016B 319      .ENABL  LSB
    016B 320
    016B 321      ; Setup fib fields
    016B 322
    016B 323
```



```

      51 01F4 CA 9E 016B 324
14 A1 0100 8F B0 0170 325 SETFIB: MOVAB FFAST_FIBBUF(R10),R1 ; fib address
10 AA 00000040 8F D0 0176 326 MOVW #FIBSW_WILD,FIBSW_NMCTL(R1) ; set wildcarding on
      14 AA 51 D0 017E 327 MOVL #FIBSC_LENGTH,FWASQ_FIB(R10) ; create fib descriptor
      3C 0182 328 MOVL R1,FWASQ_FIB+4(R10)
      0170 CA 012E 8F 3C 0182 329 MOVZWL #FWASS_NAMEBUF+-FWASS_TYPEBUF+FWASS_VERBUF,-FWASQ_NAME(R10) ; set length of result buffer
      0183 330
      0183 331
      0189 332
      0189 333
      0189 334 ; If remove and the nam fop bit is set, set fib bit to do
      0189 335 ; find via fid rather than by name.
      0189 336
      0189 337
      35 6E 91 0189 338 CMPB (SP),#IOS_DELETE ; remove function?
      0F 12 018C 339 BNEQ 20$ ; branch if not
0A 04 A8 18 E1 018E 340 BBC #FIBSV_NAM,FIB$FOP(R8),20$ ; branch if nam bit not set
      OA A1 D5 0193 341 TSTL FIBSW_DID(R1) ; fid supplied?
      05 13 0196 342 BEQL 20$ ; branch if not
      0198 343 SSB #FIBSV_FINDFID,FIBSW_NMCTL(R1) ; find by fid
      019D 344
      019D 345
      019D 346 ; If the directory file has already been read into virtual
      019D 347 ; memory, then skip the call to the acp and look in memory
      019D 348 ; for the next file name in sequence.
      019D 349
      019D 350
      32 6E 91 019D 351 20$: CMPB (SP),#IOS_ACCESS ; access function?
      13 12 01A0 352 BNEQ 22$ ; only on searches
      57 30 AA D0 01A2 353 MOVL FWASL_DIRBDB(R10),R7 ; is there a directory in memory?
      0D 13 01A6 354 BEQL 22$ ; call acp if not
      52 0188 CA 7D 01A8 355 MOVQ FWASQ_RNS(R10),R2 ; pass descriptor of file name
      00000000 EF 16 01AD 356 JSB RMSDIRSCAN ; find the next find in sequence
      13 11 01B3 357 BRB 24$ ; re-join after acp call
      01B5 358
      01B5 359
      01B5 360 ; Call acp for next file in this directory
      01B5 361
      01B5 362
      50 6E D0 01B5 363 22$: MOVL (SP),R0 ; get acp function code
      7E 7C 01B8 364 CLRQ -(SP) ; p5/p6 = 0
      0170 CA 9F 01BA 365 PUSHAB FWASQ_NAME(R10) ; p4 = result descriptor
      6C A9 9F 01BE 366 ; also input to acp as previous
      01BE 367 ; position (file) in directory
      01C1 368 PUSHAB IFB$RNS_LEN(R9) ; p3 = longword to receive length
      01C1 369 ; also input to acp as previous
      01C1 370 ; position (file) in directory
      0188 CA 9F 01C1 371 PUSHAB FWASQ_RNS(R10) ; p2 = name descriptor
      FE38 30 01C5 372 BSBW RMSFCPENC ; call acp and wait for reply
      07 50 E9 01C8 373 24$: BLBC R0,ACPERR ; branch if error from acp
      00B7 31 01CB 374 SSB #IFB$V_FILEFOUND,(R9) ; indicate at least one file found
      01D2 375 BRW COPY_RESULT ; and copy result string
      21 6A 1C E1 01D2 376
      0910 8F 50 B1 01D6 377 ACPERR: BBC #FWASV_WILD_DIR,(R10),25$ ; if there are no wild directories
      24 13 01DB 378 ; report fnf if none were
      379 ; no files in directory at all?
      380 ; if so, get next directory
      379 CMPW R0,#SS$NOSUCHFILE
      380 BEQL NEXT_DIR
```

```
0930 8F 50 B1 01DD 381 CMPW RO,#SS$ NOMOREFILES ; no more files in directory?
      1D 13 01E2 382 BEQL NEXT_DIR ; if so, get next directory
      01FE CA B4 01E4 383 CLRW FFAST_FIBBUF+FIB$W_DID(R10) ; mark fresh directory needed
51 4C AA D0 01E8 384 MOVL FFAST_SWB_PTR(R10),R1 ; get SWB ptr
      01EC 385 SSB #SWB$V TRAVERSE,- ; set to skip rest of subtree
      01EC 386 SWB$B FLAGS(R1)
0828 8F 50 B1 01F0 387 CMPW RO,#SS$ BADIRECTORY ; bad directory format?
      0A 13 01F5 388 BEQL NEXT_DIR ; ignore bad directories on traversa
      FE01' 30 01F7 389 25$: RMSERR FND,R1 ; set default error
      1C 11 01FC 390 BSBW RMSMAPERR ; map error to rms error
      01FF 391 BRB ERROR ; process other type of error
      0201 392
      0201 393
      0201 394 : If no more files in directory, skip to next directory
      0201 395
      0201 396
      0201 397 NEXT_DIR:
      0168 30 0201 398 BSBW RETDIRBDB ; deallocate directory buffer
69 04 E0 0204 399 BBS #DEV$V_SDI,IFB$S_PRIM_DEV(R9),- ; nmf if sdi device
      10 0207 400 ERRNMF
00000000'EF 16 0208 401 JSB RM$NEXTDIR ; get next directory
      0C 50 E9 020E 402 BLBC RO,ERROR ; if error, copy result and exit
      0204 CA D4 0211 403 CLRL FFAST_FIBBUF+FIB$S_WCC(R10) ; start at 1st file in directory
      FEBE 31 0215 404 BRW READ_DIR ; and then get next file
      0218 405
      0218 406 ERRNMF: RMSERR NMF ; no more files
      021D 407
      021D 408
      021D 409 : If there is no wild card directory and the user did not specify NAM$V_SVCTX,
      021D 410 then the ACP is maintaining context and we should just return the error
      021D 411 it gave us.
      021D 412
      021D 413 : If we are maintaining context (wild directory or NAM$V_SVCTX), then we should
      021D 414 convert NMF to FNF based on FILEFOUND bit.
      021D 415
      021D 416
      4F 69 39 E1 021D 417 ERROR: BBC #IFB$V_SEARCH,(R9),35$ ; we are not keeping context
      4B 69 3C E0 0221 418 BBS #IFB$V_FILEFOUND,(R9),35$ ; skip if file found
82CA 8F 50 B1 0225 419 CMPW RO,#RMS$_NMF&^XFFFF ; and error was NMF
      44 12 022A 420 BNEQ 35$
      022C 421
      022C 422
      022C 423 : If there was a wild directory, move the expanded name string from
      022C 424 the namblk to the resultant name string and return file not found
      022C 425
      022C 426
      40 6A 1C E1 022C 427 BBC #FWASV_WILD_DIR,(R10),35$ ; branch if dir not wild
58 24 A9 D0 0230 428 MOVL IFB$S_LAST_FAB(R9),R8 ; get the last fab's addr
57 28 A8 D0 0234 429 MOVL FAB$S_NAM(R8),R7 ; get the name block addr
      FDC5' 30 0238 430 BSBW RM$CHRNAM ; check nam validity
      69 50 E9 023B 431 BLBC RO,44$ ; quit on failure
      03 A7 94 023E 432 CLRB NAM$B_RSL(R7) ; assume can't set result string
52 02 A7 9A 0241 433 MOVZBL NAM$B_RSS(R7),R2 ; get length of resultant buffer
53 04 B7 DE 0245 434 MOVAL @NAM$C_RSA(R7),R3 ; get addr of resultant buffer
      0249 435 IFNOWRT R2,(R3),50$ ; probe the resultant string buff
      024F 436 ; error if can't write it
52 0A A7 9A 024F 437 MOVZBL NAM$B_ESS(R7),R2 ; get the buffer size into longword
```

```
51  OC B7  DE 0253 438      MOVAL  @NAMS$ ESA(R7),R1      ; get addr of expanded buffer
                                0257 439      IFNORD  R2,(R1),50$      ; probe the expanded string buff
                                025D 440      ; error if can't read it
52  OB A7  9A 025D 441      MOVZBL  NAMS$ ESL(R7),R2      ; get the string's actual length
03  A7 52 90 0261 442      MOV      R2,NAMS$ RSL(R7)      ; stuff the resultant length
63  61 52 28 0265 443      MOV      R2,(R1),R3      ; move the expanded string
                                0269 444      ; to the resultant string
                                0269 445      RMSERR  FNF      ; restore the error
                                026E 446      BRB      50$      ; and continue
                                0270 447
                                0270 448
                                0270 449      ; Error has occurred - setup file name so that when result
                                0270 450      ; name string is copied, the file string sent to acp is returned.
                                0270 451
                                0270 452
15  6A 19  E0 0270 453 35$:  BBS      #FWASV_NODE,(R10),COPY_RESULT ; branch if network operation
                                0274 454      PUSH      R0      ; save status code
6C  A9 0188 CA 3C 0276 455      MOVZWL  FWASQ_RNS(R10),IFB$ LNS LEN(R9) ; set length of string
018C DA 0188 CA 28 027C 456      MOV      FWASQ_RNS(R10),@FWASQ_RNS+4(R10),-
                                0283 457      FFAST_NAMEBUF(R10)
                                0286 458      POPL      R0      ; restore status
                                0289 459
                                0289 460      ;
                                0289 461      ; Copy result file name to user result buffer
                                0289 462      ; unless no file was found
                                0289 463
                                0289 464
                                0289 465 COPY_RESULT:
58  24 A9  D0 0289 466      PUSH      R0      ; save status code
57  28 A8  D0 028B 467      MOVL      IFB$ LAST_FAB(R9),R8      ; get fab address
                                028F 468      MOVL      FAB$ NAM(R8),R7      ; set nam address
                                0293 469      BSBW      RMS$CHNAM      ; check nam validity
                                0296 470      BLBC      R0,42$      ; quit on failure
                                0299 471      BSBW      RMS$COPY_RESULT      ; copy result name string
                                029C 472      BLBC      R0,42$      ; branch if error
                                029F 473      POPL      R0      ; restore status code
                                02A2 474      BRB      50$      ; and continue
SE  04 C0 02A4 475 42$:  ADDL      #4,SP      ; ignore saved status code
                                02A7 476 44$:  BRB      EXIT      ; and report one from copy_result
                                02A9 477
                                02A9 478      ;
                                02A9 479      ; If not remove, copy fid and did into nam block
                                02A9 480
                                02A9 481
                                02A9 482 50$:  CMPB      (SP),#IOS_DELETE      ; remove function?
                                02AC 483      BEQL      60$      ; if so, skip this
UC  6A 19  E0 02AE 484      BBS      #FWASV_NODE,(R10),60$      ; skip also if network operation
                                02B2 485
                                02B2 486      ASSUME  FIBSW_DID      EQ      FIBSW_FID+6
                                02B2 487      ASSUME  NAMSW_DID      EQ      NAMSW_FID+6
                                02B2 488
24  A7 01F8 CA 7D 02B2 489      MOVQ     FFAST_FIBBUF+FIBSW_FID(R10),NAMSW_FID(R7)
2C  A7 0200 CA D0 02B8 490      MOVL     FFAST_FIBBUF+FIBSW_FID+8(R10),NAMSW_FID+8(R7)
                                02BE 491
                                02BE 492      ;
                                02BE 493      ; If this is a temporary ifab/fwa created for this call
                                02BE 494      ; only, then save the current acp position in the directory
```



```
02BE 495 ; file and cleanup all internal structures.
02BE 496 ;
02BE 497 ;
0B 69 39 E0 02BE 498 60$: BBS #IFBSV_SEARCH,(R9),65$ ; branch if ifab to be saved
26 50 E9 02C2 499 BLBC R0,NMF ; go set NMF bit if any error
0204 CA 3C 02C5 500 MOVZWL FFAST_FIBBUF+FIBSL_WCC(R10),- ; save acp position
30 A7 02C9 501 NAMSL_WCC(R7)
28 11 02CB 502 BRB EXIT
02CD 503 ;
02CD 504 ;
02CD 505 ; This is a permanent ifab/fwa (that is, it is kept around between
02CD 506 calls in order to speed up things or keep extended context)
02CD 507 ; If the status was successful or not enough privilege,
02CD 508 ; then keep the wildcard sequence context around so that
02CD 509 ; search can be called again. else, cleanup everything.
02CD 510 ;
02CD 511 ;
02CD 512 65$: BLBS R0,70$ ; continue sequence if successful
82CA 8F 50 B1 02D0 513 CMPW R0,#RMS$_NMF&^XFFFF ; done with wildcard sequence?
21 13 02D5 514 BEQL CHKLST ; if so, terminate sequence
8292 8F 50 B1 02D7 515 CMPW R0,#RMS$_FNF&^XFFFF ; file not found?
1A 13 02DC 516 BEQL CHKLST ; if so, terminate sequence
0B 6A 1C E1 02DE 517 BBC #FWASV_WILD DIR,(R10),NMF ; if nonwild, cleanup
0C A8 D5 02E2 518 TSTL FASL_STV(R8) ; error from acp?
06 13 02E5 519 BEQL NMF ; if not, terminate sequence
02 A8 B4 02E7 520 70$: CLRW FASW_IF1(R8) ; mbz for subsequent operations on f
FD13' 31 02EA 521 BRW RMSEXRMS ; exit without cleaning up
30 A7 40000000 8F D0 02ED 522 ;
02F5 523 NMF: MOVL #NAMSM_SRCHNMF,NAMSL_WCC(R7) ; indicate that another search isn't
FD08' 31 02F5 524 ; to be done with this NAM
02F8 525 EXIT: BRW RMSCLSU ; cleanup ifab and buffers
02F8 526 ;
02F8 527 .DSABL LSB
02F8 528 ;
02F8 529 ;
02F8 530 ; We are about to exit with No More Files or File Not Found, before we
02F8 531 ; really do, check to see if there was a search list, if so try to
02F8 532 ; parse a new string and if successful search for a new file
02F8 533 ;
02F8 534 ;
F1 6A 38 E1 02F8 535 CHKLST: BBC #FWASV_SLPRESENT,(R10),NMF ; are search list present?
02FC 536 SSB #FWASV_SL_PASS,(R10) ; indicate search list parse
30 A7 DD 0300 537 PUSHL NAMSL_QCCT(R7) ; save wild card context
57 DD 0303 538 PUSHL R7 ; save NAM blk ptr
FCF8' 30 0305 539 BSBW RMSPARSE_FILE ; parse a new string
57 8ED0 0308 540 POPL R7 ; restore NAM blk ptr
30 A7 8ED0 030B 541 POPL NAMSL_WCC(R7) ; restore wcc
03 50 E9 030F 542 BLBC R0,10$ ; branch if error
FD66 31 0312 543 BRW SRCH ; go search new string
0315 544 ;
0315 545 ;
0315 546 ; If there was a file found on some previous search operation then
0315 547 ; convert DNF and FNF errors into NMF
0315 548 ;
0315 549 ;
D4 69 3C E1 0315 550 10$: BBC #IFBSV_FILEFOUND,(R9),NMF ; no previous file found
C04A 8F 50 B1 0319 551 CMPW R0,#RMS$_DNF&^XFFFF ; directory not found
```

RMSOSRCH
V04-000

SEARCH FOR NEXT WILDCARD FILE
RMS\$SEARCH, Search for next Filename in

1 2

16-SEP-1984 01:32:07
5-SEP-1984 16:25:32

VAX/VMS Macro V04-00
[RMS.SRC]RMSOSRCH.MAR;1

Page 13
(5)

8292 8F	07	13	031E	552	BEQL	20\$: yes, convert it
	50	B1	0320	553	CMPW	RO, #RMS\$_FNF&^XFFFF		: file not found
	C6	12	0325	554	BNEQ	NMF		: no exit
			0327	555	RMSERR	NMF		: convert the error
	BF	11	032C	556	BRB	NMF		: exit
			032E	557				

```
032E 559 .SBTTL RMSCOPY_RESULT, Return Result Name String
032E 560
032E 561 :++
032E 562
032E 563 RMSCOPY_RESULT
032E 564
032E 565 Construct the result name string and return to
032E 566 the caller via the rsa and rss fields of the nam.
032E 567
032E 568 inputs:
032E 569
032E 570 r7 = address of NAM
032E 571 r9 = address of ifab
032E 572 r10 = address of fwa
032E 573 ifb$len_rns_len = length of new file name
032E 574 fwa$st_namebuf = new file name string
032E 575 fwa$q_device = descriptor of device name
032E 576 fwa$q_dir1-8 = descriptors of directory names
032E 577 fwa$b_dirlen = number of directory levels
032E 578 fwa$b_dirterm = directory specification terminator
032E 579
032E 580 outputs:
032E 581
032E 582 result string buffer is output if requested.
032E 583 NAMSL_FNB
032E 584
032E 585 :--
032E 586
032E 587 RMSCOPY_RESULT::
032E 588 BBS #FWASV_NODE, (R10), 5$ ; branch if network operation
032E 589 MOVL IFB$LEN_RNS_LEN(R9), - ; set length of file name
032E 590 FWA$Q_NAME(R10)
032E 591 BBC #FWASV_DIR, (R10), 5$ ; skip if no directory in spec
032E 592 BBC #FWASV_WILD_DIR, (R10), 5$ ; or if there are no wild directori
032E 593 SUBB3 #1, FWA$B_DIRLEN(R10), R0 ; get number of subdirectory levels
032E 594 INSV R0, #FWASV_DIR_LVL, - ; return current # of subdir.
032E 595 #FWASV_DIR_LVL, (R10) ; levels in the FWA
032E 596 INSV R0, #NAMSV_DIR_LVL, - ; return current # of subdir.
032E 597 #NAMSV_DIR_LVL, - ; levels in the NAM
032E 598 NAMSL_FNB(R7)
032E 599 MOVB FWA$B_DIRWCF_LGS(R10), - ; if any ellipses were found,
032E 600 NAMSL_FNB+3(R7) ; set the appropriate wild
032E 601 ; flags in the nam blk
032E 602 5$: MOVAB B*10$, AP ; address of expstring arg list
032E 603 PUSHL NAMSW_FID(R7) ; save contents of nam fid
032E 604 CLRL NAMSW_FID(R7) ; clear fid so expstring will work
032E 605 BSBW RMSEXPSTRING ; return result name string
032E 606 POPL NAMSW_FID(R7) ; restore contents of nam fid
032E 607 RSB
032E 608
032E 609 10$: .BYTE NAMSL_RSA ; offset to result buffer addr.
032E 610 RMSERR_WORD RST ; error of bad buffer
032E 611 RMSERR_WORD RSS ; error of buffer too short
```

23 6A 19
6C A9
0170 CA
19 6A 0E
15 6A 1C E1
2E AA 01 B3
1D 50 F0
6A 03
15 50 F0
03
34 A7
05 AA 90
37 A7
5C 67 AF 9E
24 A7 DD
24 A7 D4
FC9E 30
24 A7 8ED0
05
04


```
036C 613 .SBTTL RETDIRBDB, Deallocate Directory Buffer and BDB
036C 614
036C 615 :++
036C 616
036C 617 RETDIRBDB
036C 618
036C 619 This routine deallocates the directory buffer and the bdb
036C 620 which is associated with it.
036C 621
036C 622 inputs:
036C 623
036C 624 r10 = fwa address
036C 625 r9 = ifab address
036C 626 fwa$l_dirbdb = address of directory bdb
036C 627
036C 628 outputs:
036C 629
036C 630 none
036C 631 --
036C 632
036C 633
036C 634 RETDIRBDB:
54 30 AA D0 036C 635 MOVL FWA$L_DIRBDB(R10),R4 ; is there a directory in memory?
OE 13 0370 636 BEQL 10$ ; branch if not
SA 59 DD 0372 637 PUSHL R10 ; save r10
FC86' 30 0374 638 MOVL R9,R10 ; rm$retbdb wants ifb address in r10
SA 8ED0 0377 639 BSBW RM$RETBDB ; deallocate it if there is
30 AA D4 037A 640 POPL R10 ; restore r10
05 037D 641 CLRL FWA$L_DIRBDB(R10) ; and clear pointer
0380 642 10$: RSB
0381 643
0381 644 .END
```

RMSOSRCH
Symbol table

SEARCH FOR NEXT WILDCARD FILE

L 2

16-SEP-1984 01:32:07 VAX/VMS Macro V04-00
5-SEP-1984 16:25:32 [RMS.SRC]RMSOSRCH.MAR;1

Page 16
(7)

\$.PSECT EP	= 00000000			FWASV_WILD DIR	= 0000001C		
\$\$RMSTEST	= 0000001A			IFBSL_AS_DEV	= 0000008C		
\$\$RMS_PBUGCHK	= 00000010			IFBSL_FWA_PTR	= 00000038		
\$\$RMS_TBUGCHK	= 00000008			IFBSL_LAST_FAB	= 00000024		
\$\$RMS_UMODE	= 00000004			IFBSL-PRIM-DEV	= 00000000		
ACPERR	000001D2	R	01	IFBSL-RNS_CEN	= 0000006C		
CHKLST	000002F8	R	01	IFBSV_ACCESSED	= 00000025		
COMMON	00000008	R	01	IFBSV-BUSY	= 00000020		
COPY_RESULT	00000289	R	01	IFBSV_FILEFOUND	= 0000003C		
DEVSV_DIR	= 00000003			IFBSV-NSP	= 0000003F		
DEVSV_SDI	= 00000004			IFBSV-SEARCH	= 00000039		
DEVSV_SPL	= 00000006			IMPSL-MAVED-SP	= 00000014		
ENS	0000009F	R	01	IOS_ACCESS	= 00000032		
ERRESA	0000015C	R	01	IOS_DELETE	= 00000035		
ERRESL	00000163	R	01	NAMSB_ESL	= 0000000B		
ERRIOP	00000092	R	01	NAMSB-ESS	= 0000000A		
ERRNMF	00000218	R	01	NAMSB-RSL	= 00000003		
ERROR	0000021D	R	01	NAMSB-RSS	= 00000002		
ERRWCC	0000009A	R	01	NAMSL-ESA	= 0000000C		
EXIT	000002F5	R	01	NAMSL-FNB	= 00000034		
EXIT1	00000097	R	01	NAMSL-RSA	= 00000004		
EXIT2	00000168	R	01	NAMSL-WCC	= 00000030		
FABSL_FOP	= 00000004			NAMSM-SRCHNMF	= 40000000		
FABSL_NAM	= 00000028			NAMSM-SVCTX	= 80000000		
FABSL-STV	= 0000000C			NAMSS-DIR-LVLS	= 00000003		
FABSV_NAM	= 00000018			NAMSV-DIR-LVLS	= 00000015		
FABSW_IFI	= 00000002			NAMSV-IFI	= 00000010		
FIBSC_LENGTH	= 00000040			NAMSV-SRCHNMF	= 0000001E		
FIBSL_WCC	= 00000010			NAMSW-DID	= 0000002A		
FIBSM_WILD	= 00000100			NAMSW-FID	= 00000024		
FIBSV_FINDFID	= 0000000B			NEXT_DIR	00000201	R	01
FIBSW-DID	= 0000000A			NEXT_FILE	000000F4	R	01
FIBSW-FID	= 00000004			NMF	000002ED	R	01
FIBSW-NMCTL	= 00000014			NT\$ACCESS	*****	X	01
FSCBSC_BLN	= 00000104			NT\$REMOVE	*****	X	01
FSCBSQ_NAME	= 0000002C			NT\$SEARCH	*****	X	01
FSCBSQ_TYPE	= 00000034			NTSRCH	000000A2	R	01
FSCBSQ_VERSION	= 0000003C			PIOSA TRACE	*****	X	01
FWASB_DIRLEN	= 0000002E			READ DIR	000000D6	R	01
FWASB_DIRWCFLGS	= 00000005			RETDIRBDB	0000036C	R	01
FWASL_DIRBDB	= 00000030			RMSCHKNAM	*****	X	01
FWASL-SWB_PTR	= 0000004C			RMSCLSCU	*****	X	01
FWASQ_FIB	= 00000010			RMSCOPY_RESULT	0000032E	RG	01
FWASQ_NAME	= 00000170			RMSDIRSCAN	*****	X	01
FWASQ-RNS	= 00000188			RMSXPSTRING	*****	X	01
FWASS-DIR LVLS	= 00000003			RMSXRMS	*****	X	01
FWASS-NAMEBUF	= 00000100			RMSX NOSTR	*****	X	01
FWASS-TYPEBUF	= 00000028			RMSFABCHK	*****	X	01
FWASS-VERBUF	= 00000006			RMSFCPFNC	*****	X	01
FWAST_FIBBUF	= 000001F4			RMSFSETI ALT	*****	X	01
FWAST-NAMEBUF	= 000004B6			RMSFSET ALT1	*****	X	01
FWASV_DIR	= 0000000E			RMSGETSPC1	*****	X	01
FWASV-DIR LVLS	= 0000001D			RMSMAPERR	*****	X	01
FWASV-NODE	= 00000019			RMSNEXTDIR	*****	X	01
FWASV-SLPRESENT	= 00000038			RMSPARSE_FILE	*****	X	01
FWASV-SL_PASS	= 00000002			RMSREADDIR	*****	X	01
FWASV-WILDCARD	= 00000018			RMSRECOVER_FWA	*****	X	01

RMSOSRCH
Symbol table

SEARCH FOR NEXT WILDCARD FILE

M 2

16-SEP-1984 01:32:07 VAX/VMS Macro V04-00
5-SEP-1984 16:25:32 [RMS.SRC]RMSOSRCH.MAR;1

Page 17
(7)

```
RMSRETBDB          ***** X 01
RMSRETSPC1         ***** X 01
RMSSCAN STRING     ***** X 01
RMS$REMOVE         = FFFFFFFE RG 01
RMS$SEARCH         = 00000003 RG 01
RMS$DNF            = 0001C04A
RMS$ESA            = 000184FC
RMS$ESL            = 00018714
RMS$FND            = 0001C02A
RMS$FNF            = 00018292
RMS$IFI            = 00018564
RMS$IOP            = 00018574
RMS$NMF            = 000182CA
RMS$RSS            = 00018694
RMS$RST            = 0001869C
RMS$WCC            = 000182EA
SETFTB             0000016B R 01
SRCH               0000007B R 01
SS$BADIRECTORY     = 00000828
SS$NOMOREFILES     = 00000930
SS$NOSUCHFILE      = 00000910
SWB$B_FLAGS        = 00000000
SWB$V_TRAVERSE     = 00000004
TPT$SL_SEARCH      ***** X 01
```

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes														
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
RMSRMS	00000381 (897.)	01 (1.)	PIC	USR	CON	REL	GBL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE				
SAB\$S	00000000 (0.)	02 (2.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE				

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.12	00:00:00.72
Command processing	109	00:00:00.70	00:00:04.02
Pass 1	522	00:00:21.26	00:00:53.49
Symbol table sort	0	00:00:03.48	00:00:05.19
Pass 2	130	00:00:03.92	00:00:09.10
Symbol table output	17	00:00:00.16	00:00:00.56
Psect synopsis output	1	00:00:00.02	00:00:00.09
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	810	00:00:29.66	00:01:13.18

The working set limit was 1800 pages.
119924 bytes (235 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2357 non-local and 34 local symbols.
644 source lines were read in Pass 1, producing 15 object records in Pass 2.
32 pages of virtual memory were used to define 31 macros.

! Macro library statistics !

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	16
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	3
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	27

2488 GETS were required to define 27 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:RMSOSRCH/OBJ=OBJ\$:RMSOSRCH MSRC\$:RMSOSRCH/UPDATE=(ENH\$:RMSOSRCH)+EXECML\$/LIB+LIB\$:RMS/LIB

0331

AH-BT13A-SE
 VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY